第 40 卷 第 2 期
2022 年 6 月
天 文 学 进 展
PROGRESS IN ASTRONOMY
Vol. 40, No. 2
June., 2022

doi: 10.3969/j.issn.1000-8349.2022.02.10

基于 MongoDB 的海量天文星表数据的快速时序重构研究

徐丹滢，赵 青，权文利，宋红壮

(天津科技大学 人工智能学院，天津 300457)

摘要： 天文数据的爆发性增长，导致运用传统科学计算方法生成天文时序数据时效率不高，直接影响时域天文学的科学产出。为了解决这一问题，文章提出了减少距离计算的同源星表快速证认方法及基于 MongoDB 的应用方案，重点从原始数据的访存优化，证认计算速度的提高等方面寻求新的改进方案，以解决大规模天文星表的批量时序重构的效率问题。实验结果表明，与基于传统多波段交叉证认算法和关系型数据库的方法相比，该方法可以更有效地提高时序数据的生成效率，为时域天文学时代频繁采样望远镜大规模星表数据的时序重构和光变曲线的生成提供了新思路。

关 键 词： MongoDB；地理空间索引；访存优化；证认优化

中图分类号： P129 **文献标识码：** A

1 引 言

近年来，“时域天文学”引起了天文学界的广泛关注，如何研发适合时域天文学的高性能天文海量数据处理软件，已经成为各国天文信息学研究的焦点。在已定标的星表上通过星体相互的交叉证认来确定每一个天体在不同时间点的数据，并按照时间进行排序生成每个天体的时序数据，这一过程称为星表数据的时序重构。它是时域天文学研究中的一个重要数据处理步骤，也是拟合光变曲线，开展时域分析研究的基础。

面向大规模星表数据的高效时序重构处理方法可以有效提高天文时序数据产品的生成效率，促进时域天文学研究的快速发展。但目前大规模星表的时序重构仍需要攻破长时区间上多星表间批量交叉证认的计算难题，传统交叉证认方法并不能满足时序重构问题的效率

收稿日期： 2021-08-17； **修回日期：** 2021-11-25

资助项目： 国家自然科学基金青年项目 (11803022)；天津市科委自然科学基金青年项目 (18JCQNJC69800)；天津市教委科研计划项目 (2017KJ035, 2018KJ105)；天津市企业科技特派员项目 (20YDTPJC00560)

通讯作者： 赵青，zhaoqing@tust.edu.cn

要求，需要更进一步的算法优化。在传统的异源星表多波段交叉证认方法中，由于复杂的距离计算和边缘数据的额外处理需要，计算效率较低，而对于同源星表的时序重构，相较于异源问题更面临着一个重大的难点：要进行长时区间上的反复多次交叉证认，传统交叉证认方法不足以满足时序重构的效率要求，需要进一步的算法优化。

近年来，在海量天文星表数据的检索、存储与融合等方面，很多学者已进行了深入的研究。高丹等人[1, 2]提出了基于 HTM 索引和 KD-Tree 的交叉证认方法，但采用了关系型数据库模式，只能支持几十万条数据的中等规模星表，而且不能避免分块边缘的漏源问题。Zhao 等人[3, 4]提出了基于 MPI 的多核并行交叉证认算法，通过基于位运算的快速相邻块编码推导算法高效地取得误差半径范围内全部需证认的数据，解决了边缘漏源问题，但在数据库选择上采用了 MySQL 传统关系型数据库，效率上仍然无法满足批量多星表间的证认需求。徐洋等人[5, 6]提出基于等经纬分区二维空间网格的方法，但因其特殊的分块方式，仅适用于 100 ~ 200 平方度的中等天区覆盖大小的数据。Du 等人[7, 8]则采用了基于 HTM 和 HEALPix 两种分区的混合证认计算算法来解决星表数据的漏源问题，虽然大大减少了漏源现象的产生，但预处理时间长，在交叉证认计算过程中也引入了重复计算量，且采用了 SQL Server 关系型数据库，只能在 Windows 平台上运行，其并行实施和共存模型也难以处理日益增多的数据需求。

而对于大规模星表数据来说，非关系型数据库灵活的模型、丰富的功能、非结构化的存储及高扩展性，使得它更适用于数据处理，也更有望提高海量天文数据重构计算方面的性能。目前越来越多的人发现了非关系型数据库的优势，万萌[9]针对 GWAC 数据管理提出了基于 MonetDB 数据库的处理方案，基于该生成器的模拟数据，使用 MonetDB 数据库内置 SQL 实现了交叉认证计算。MonetDB 数据库较传统关系型数据库进行了适合于科学数据计算特点的优化，取得了较好的效果。但此方法在规模扩展上仍存在一定的问题，比如实验已表明[10]，对于大规模累积数据来说，MonetDB 的支持性不足，且入库时间不够稳定，容易造成数据拥塞。冯磊等人[11]利用非关系型数据库 MongoDB 存储航磁大数据，证明了 MongoDB 在数据写入性能及空间查询效率方面表现非常优异。程江林[12]基于 MongoDB 设计了灵活的数据库结构，并采用分片技术部署 MongoDB 集群实现水平扩展来存储数据。周尧等人[13]提出了基于 Spark 和 MongoDB 的地理空间数据的分析方案，利用 MongoDB 的地理空间索引和查询机制来处理地理空间数据。这些实验表明 MongoDB 数据库不仅查询速度快，支持分布式集群，且对处理高并发、大批量的地理空间数据十分合适。

此外，交叉证认在同源星表上也有研究。如 Li 等人[14]提出了一种并行环境下的同源星表证认优化算法，并使用动态规划思想分割数据，以保证并行环境下的负载平衡，其研究成果被应用于生成星表的光变曲线，可以描述出天体的亮度随时间的变化。但他们仍采用基于 HTM 和 HEALPix 两种分区的混合证认计算方式来解决边界漏源问题，重复计算使得计算量较大，他们采用的并行环境也不适合规模扩展，难以应付数据量的进一步扩大。

虽然当前传统交叉证认算法已经得到了多年的研究，方法已经较为成熟，但考虑到时域天文学中望远镜的高采样频率，时序证认需要在长时间轴上进行连续星表间的反复证认，对效率要求更高，必须要更进一步在存储访问和证认计算简化上下功夫。本文研究的是同源

星表，具有一致的望远镜系统误差，星体的位置变动极小，只在一定的局部地区有少量的共有误差，不需要全部星体都进行严格的距离证认，从而可以考虑在一定程度上减少距离计算量，优化算法的复杂度，提高效率，使大规模批量同源星表间的高效证认成为可能。

综合以上问题，本文针对天文星表数据的海量性，考虑到同源星表区别于异源星表的特点，提出了在 MongoDB 环境下面向海量同源星表数据的高性能时序数据重构算法。而大量星表证认的效率问题主要体现在两个方面：1) 数据存储访问效率方面，针对这个问题本文利用了 MongoDB 的文档式分片集群存储和地理空间索引来实现数据的访存优化；2) 计算优化方面，本文提出减少距离依赖的快速交叉证认算法，以优化算法的复杂度，并与传统基于距离计算的方法进行对比，通过特定区域的范围过滤、局部证认计算提高算法的精度。对于边界漏源的处理也只以第一天数据为参考星表，加入冗余数据，后面则不再需要冗余数据，从而提高效率，也有利于数据的分布式划分，减少节点间的数据通信量。其算法的整体设计框架图如图 1 所示。

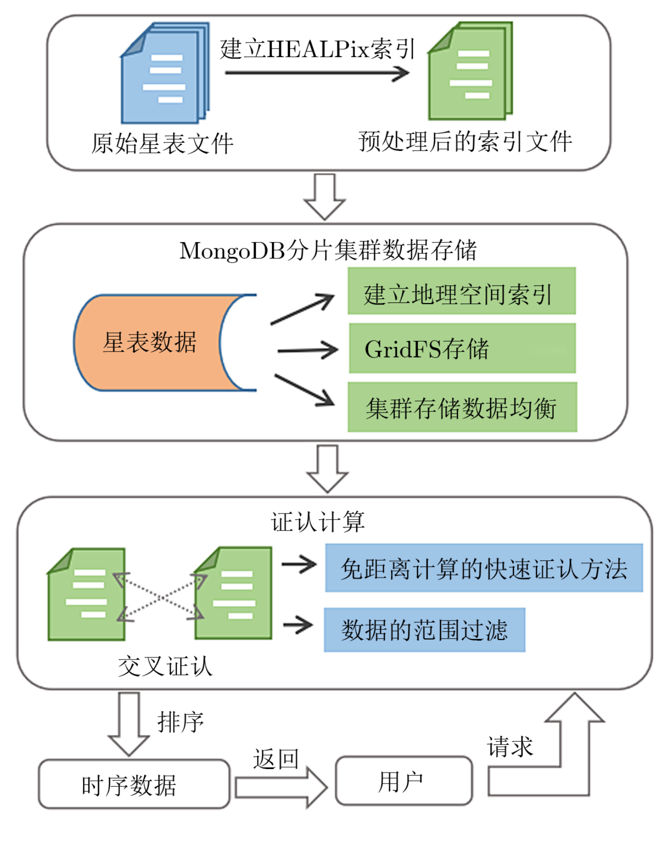


图 1　整体框架图

2　星表数据的快速交叉证认算法

交叉证认算法是时序重构的重要一环，目前关于交叉证认的算法都是采用对星表中每一个星体都进行严格距离计算的证认方式；这种传统的证认方法都包含了复杂的三角函数运算，虽然精度很高，但存在由于计算量巨大而十分耗时的问题。且时序重构需要多星表间的批量交叉证认计算，耗时的传统交叉证认计算显然不能满足实时查询的要求，因此需要进一步优化算法。

针对此问题，本文提出了一种快速证认的算法，可以在局部范围内减少对距离计算的依赖，提高效率。本文所研究的同源星表数据在位置定标后只存在一致的系统误差，受图像对齐算法的启发，利用图像位置及亮度信息寻找目标星表和另一星表之间的最佳匹配。基于这一原理，拟在局部地区内通过位置比较、范围过滤及星等信息比较来快速定位相匹配的天体，免除大多数距离计算量，并在局部星体数量异常、星等异常等区域做严格距离计算；同时，对边缘数据进行处理，减少因漏源问题产生的错误，在优化算法的复杂度的同时保证算法的精度，提高了快速证认算法的准确率。

2.1　HEALPix 数据范围过滤策略

星表数据的数据量十分庞大，我们先对原始的星表文件进行处理，从中提取序号、赤经、赤纬、亮度等关键列，然后选取合适的天区层级并计算 HEALPix 索引，得出存储在 MongoDB 数据库中的索引表，索引表如表 1 所示。

表 1　星体索引表

字段	类型	字段描述
ID	int	Serial number
RA	Double	Right ascension
DEC	Double	Declination
HEALPIXID	Blank	HEALPix IndexID
LUMNANCE	Double	Magnitude luminance

交叉证认是通过两个星体之间的角距离来判断是否为同一星体的过程，当角距离小于阈值时才被证认为同一星体。天球上的星体一般都按照位置信息来划分，位置相距过大的星体自然不可能是同一星体，依据这一理论，我们可以用范围过滤的方法降低证认的计算量及计算次数。

范围过滤的原理如图 2 所示。证认计算首先被限定在相同 HEALPix 分区中，在已选取的 HEALPix 分区中，根据两张星体索引表，对相同分区中的星体再进行区域的限定；根据证认的误差半径要求，在所查询的星体周围形成合适的范围，过滤掉距离相差很大的星体，避免距离过大星体的无效计算，以此提高交叉证认距离计算的效率和准确率。

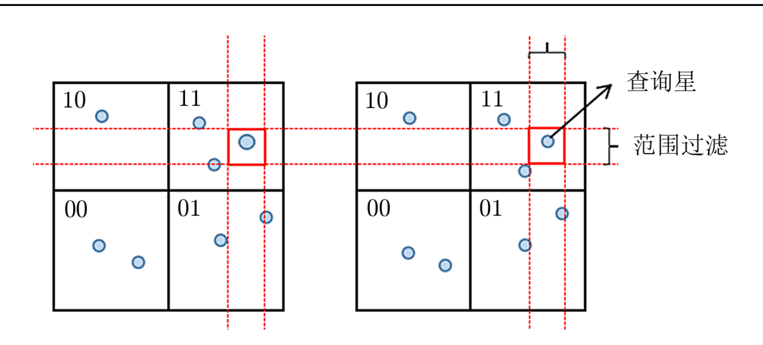


图 2 范围过滤

2.2 免距离的证认计算方式

考虑到时序重构所涉及的证认计算的对象主要为同源星表，具有一致的望远镜系统误差，受图像对齐算法的启发，可以通过局部天区星体位置以及星体亮度的比较，对星体进行免距离计算的证认，即根据误差半径选取合适的 HEALPix 层级，在该划分下落于同一块内的星体直接认为相互匹配，从而减少复杂度较高的三角函数运算。但在对比两张星表位置及亮度信息进行匹配的过程中，有可能会出现以下几种情况导致匹配错误：(1) 参考星表中某分区内的某个星体的位置接近于分区的边缘，它在待证认星表中的位置正好越过此边缘而落入相邻分区中，即发生了边界漏源现象；(2) 同一分区内的多个星体位置比较接近，赤经直接匹配导致相互混淆；(3) 星表某天区在某个时间的星表上有新星体出现。为了保证证认结果的准确率，我们针对这几种情况进行了特殊处理，如图 3 所示。

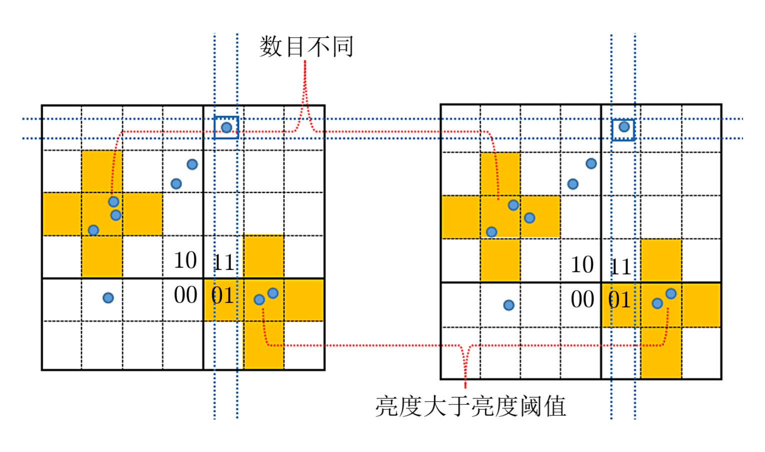


图 3 特殊情况示意图

当情况 (1) 和情况 (3) 发生时，通常表现为相同分区内星体数量不一致，我们将数量不一致的分区标记为特殊区域，为避免边界漏源则将特殊区域的周围分区也一并标记为特殊区域；当情况 (2) 发生时，我们对已经预匹配的两个星体进行星等亮度信息的比较，当超过亮度阈值时，则认为需要对这两颗星体高度关注，并将该块标记为特殊区域。被标记为特殊区域的分区内的星体将和另一张星表对应特殊区域内的星体相互进行严格距离计算的证认，

以此来尽可能避免免距离计算的误差，保证快速证认算法的准确率。

图 4 是快速证认算法的整体流程图，算法的基本原理如下：

(1) 首先根据误差半径选取细粒度的天区划分层级，以保证每个分区内的星体数目尽量均衡。

(2) 若两张星表相同分区内的星体数目相同，且星体数目为 1 时，说明此分区内除了目标星体无其他星体，可直接进行亮度的比较，若亮度小于亮度阈值，则直接通过 HEALPixID 进行匹配。

(3) 当星体数目大于 1 时，为了找出两个分区内不同星体的对应星体，首先对相同分区的星体进行范围过滤，在范围内的星体再进行亮度的比较：若亮度小于亮度阈值，则星体直接匹配；对亮度大于阈值的星体则对此星体所在的分区及其周围分区做特殊标记，经过特殊标记的分区将会和另一张参考星体的对应分区做严格的距离计算。

(4) 若相同分区内的星体数目不同，说明在此分区内可能发生了边界漏源，也就是此分区内的星体落入到其他分区内，为了解决这一问题，则对此分区及周围分区进行严格的距离计算的证认算法，以提高证认计算的准确度。

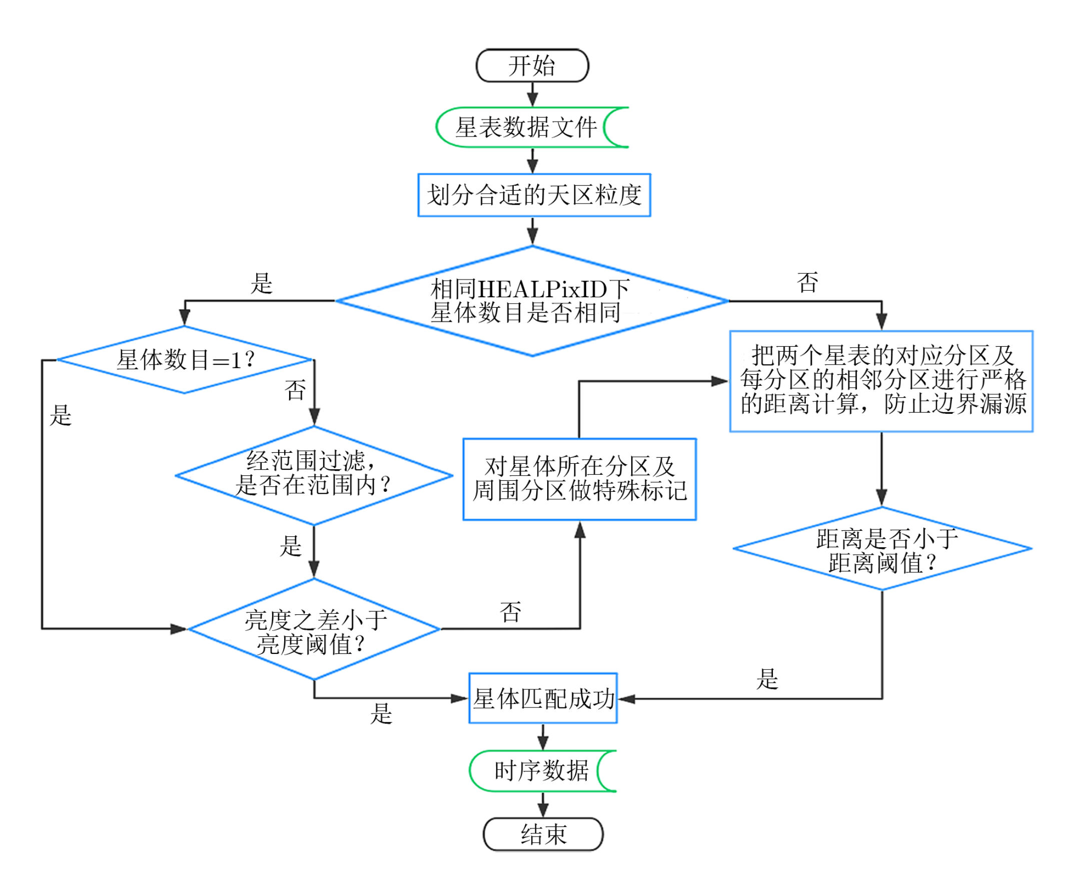


图 4　快速证认算法流程图

3　星表数据的访存优化研究

随着星表数据的爆发性增长，天文数据的访存速度在天文应用中有重要影响。而对于时序重构而言，因长时间轴上的多星表间的证认量巨大，时空数据的快速访存设计是整体性能提升的关键，数据的存储结构、索引设计更是提高访存速度的难点。

为了解决这一问题，本文选择使用 MongoDB 这一分布式的非关系型文档数据库来存储数据，如图 5 所示。MongoDB 作为最接近于传统关系型数据库的 NoSQL 数据库，兼具了既能够做到读写性能的提升又方便索引结构建立的特点，它的存储设计、以 2D 和 2Dsphere 为基础的一系列地理空间索引和查询机制、可横向扩展功能，都让它在存储天文数据时具有卓越的性能；同时，它面向数据的模型及故障自动转移能力，也让它具有高并发的读写能力以及较好的系统稳定性。MongoDB 存储的数据都是 BSON 结构，内部可以包含各种类型的文档，数据之中也可以内嵌其他数据，这种自由的存储模式也让它在面对海量科学数据的存储时，可以根据需求对每条记录来添加或者减少字段使用[15]。

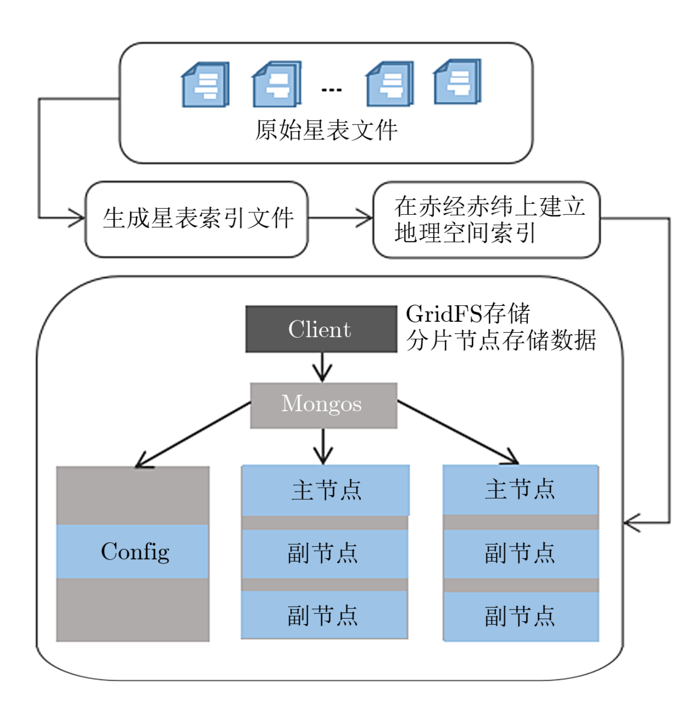


图 5　基于 MongoDB 的访存优化过程

3.1　星表数据索引研究

对于星表数据来说，索引是快速查询星表数据的关键，在多种天文数据处理和分析中都有重要作用[16]。我们常见的多维索引技术是空间多维索引，如 R-Tree, G-Tree, KD-Tree 等，但由于很难保证位置上相近的点同属于一个分支节点或叶子节点，导致上述索引在实际应用中不是很理想；且随着数据的增多，上述索引的层次也随之增多，查询效率下降，达不

到快速索引和访问的要求。而 MongoDB 的众多扩展功能之中，它的地理空间索引可以很好地支持基于空间位置的数据处理和计算。地理空间索引是一种伪二维索引，它既可以实现二维空间到一维空间的映射，又能保证相邻的区域在一维空间编码上仍然相近。

MongoDB 中的空间索引分为 2D, 2Dsphere, geoHaystack[17]。geoHaystack 是一种特殊索引，可以优化小面积内的返回结果，提高平面进行几何查询时的效率。基于键值对的地理空间索引是 2D 索引和 2Dsphere 索引，2Dsphere 索引可以理解为球面经纬度索引，支持查询球面几何实体对象；但 2Dsphere 由于是球面索引，所以仅仅支持查询经纬度数据。

2D 索引可以理解为平面 2D 索引，支持对平面地图和时间连续上的点数据索引，2D 索引 MongoDB 是以 GeoHash 技术来构建的，并没有使用国际通用的每一层 32 个 grid 的 GeoHash 的描述方式，而是采用了平面四叉树的方式。这一划分思想与天文学中的 HEALpix 索引划分思想十分相似，这种分层迭代的四叉树划分及其编码方式保证了绝大多数空间位置接近的区域编码上同样接近。2D 索引如图 6 所示。

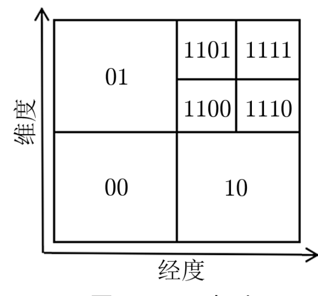


图 6　2D 索引

我们可以充分利用 MongoDB 中的地理空间索引实现对数据的快速定位，提高查询效率。根据 4.3 节的实验结果，在数据量大的星表数据中 2D 索引的效率明显高于 2Dsphere 索引，所以本文选择对星表数据建立 2D 索引，具体实验见 4.3 节。

3.2　星表数据存储研究

若想不需要大型计算机就可以处理大规模星表数据的存储和负载，数据库的可横向扩展功能非常重要。MongoDB 的分片就是对数据库的横向扩展，是将整个数据按照给定的片键分割成许多的数据块，并将这些数据块存储在不同的节点上，每个节点都维护着一个数据子集。本文利用 MongoDB 的分片功能进行数据的分布式存储，并使数据均衡分布在各个分片上，防止一个节点存储大量数据导致负载过大、搜寻数据过慢。

为了保障星表数据中的数据安全，本文用 MongoDB 副本集来将数据副本保存在多个节点上，让它在复制过程中做到数据“接近实时的同步”，从而实现高可用。副本集的结构为典型的“一主二副”架构，如图 7 所示，它的主节点和副节点都属于数据节点，都保存了完整的数据[18]。

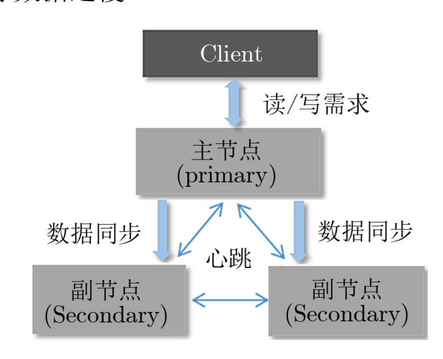


图 7　数据副本集结构

本文以“一主二副”的副本集结构作为一个分片，在机器上分别创建分片一“replcopy1”和分片二“replcopy2”，并由两个分片和一个配置节点作为一个集群。集群结构如图 8 所示。

集群搭建完成后，分片信息存放在 Config Server 中的 Shards 集合中，可以在 Mongos 端口查看分片信息。

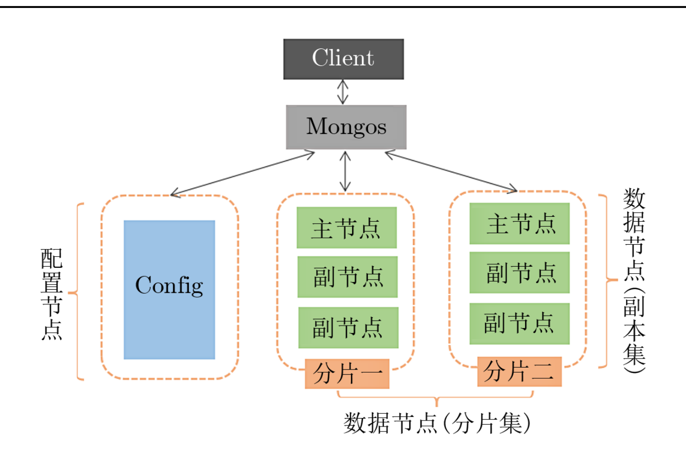


图 8　分片集群结构

3.3　存储集群数据均衡机制

集群数据的负载平衡对数据的读写效率起着至关重要的作用，很可能因为数据分布不均导致某一节点负载过大，出现失衡的情况。而 MongoDB 通过分片键对星表数据进行划分，在分片之前数据集合是一个单一的数据块，分片依据片键将集合拆分为多个数据块，这些数据块分布在不同的分片上，在对数据进行读写的过程中，片键的选择和数据平衡机制对数据分布均衡至关重要。

为了便于数据迁移和分布均衡，我们采用基于哈希 (Hash) 的分片方式。如图 9 所示，哈希分区具有天然的静态负载均衡特性，它一般追求的是数据在分区上均匀分布的特性，用户不必考虑自己指定一个列值或列值集合应该存在哪个分区上，数据库会自动完成相应的工作。因为在哈希分区中，数据都需要通过统一的哈希函数来确定存储的位置，所以当创建分区列上的数据重复率很低时，哈希分区能很好地将各个数据均匀分布在各个物理存储上。

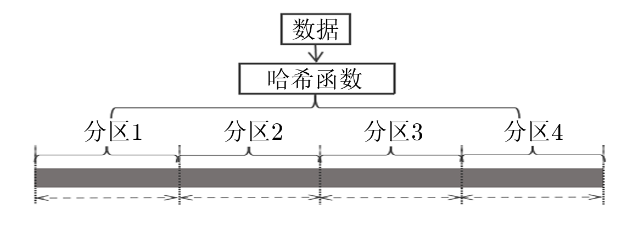


图 9　哈希分片原理

本文在星表数据的 Healpix 分区 ID 这一列 (列名为 hid) 创建哈希分片键。Healpix 层级的划分对整体星表数据存储是否均衡也起到非常关键的作用，Healpix 块划分的大小影响着每个 HealpixID 中星体的存储数目和快速证认算法的证认准确性，由 4.2.2 节的实验可知，当 Healpix 层级在 13 级时，可以保证每个 HealpixID 存储的星表数据在相对均衡的状态下，达到快速证认算法实验的效率和准确度的整体最优化。在 HealpixID 上创建哈希片键，使数据随机均匀地分布在节点上，既可以保证数据均衡性，又可以保证数据在读取时的并发性。

如果对星表数据进行实时查询，除了需要考虑节点中数据的存储平衡外，还需要考虑节点中数据访问频次的均衡。拥有热点数据的节点将会被集中访问，这时我们可以在分片设计时考虑关闭 MongoDB 的均衡器进行手动分片，或修改 MongoDB 中的均衡机制，在原本只考虑数据块量负载的基础上，再将平均数据的访问时间间隔作为一个因素考虑进去，就可以一定程度上解决数据访问频次不均衡的问题。

4　实验分析

4.1　软硬件环境

本实验的实验数据为中国虚拟天文台在网站上公开的一部分 AST3 拍摄的 HD8500 数据集。实验中，我们使用了一台 AMD Ryzen 5 3500U x8 计算机进行快速证认算法和传统证认算法的对比分析，内存为 8 GB，操作系统为 deepin15.5，编程语言为 C++；然后使用一台 Intel(R)Core(TM)i5-7200U CPU@2.50GHz 的计算机进行了 MongoDB 的读写性能测试和综合测试，内存为 8 GB，操作系统为 Windows10，编程语言为 C++ 和 python。

4.2　快速证认算法与传统证认算法的对比分析

4.2.1　融合亮度信息比较的准确率对比测试

传统的交叉证认都采用单一的基于位置的证认方法，这种方法虽然精度较高，但计算量十分巨大，考虑到本文研究的是同源星表的时序重构，对于亮度变化缓慢的长周期变源我们可以融合星体的亮度信息来进一步提高免距离计算时的效率和证认准确度。实验利用两颗星体的亮度是否小于亮度阈值来比较判断两颗星体亮度差异是否过大，从而进一步判断这两颗星体是否为同一颗星体。为了验证加入亮度信息比较的必要性，做了准确率比较实验，实验数据为 AST3 拍摄的 HD8500 数据中 a0507.2 和 a0507.3，星表数据条数分别为 37 818 条和 34 940 条，亮度阈值取值为 0.02。

图 10 为是否在快速证认算法种加入亮度信息比较的两种算法的准确率，可以看出在快速证认算法中加入星体亮度信息的比较明显提高了证认计算的准确率，使证认计算的准确率稳定在 99% 以上。

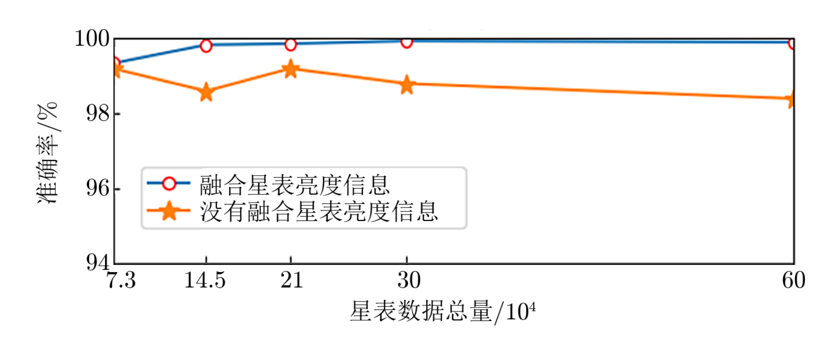


图 10　融合亮度信息的准确率测试

4.2.2 不同 HEALPix 层级下评价指标对比

交叉证认过程中，HEALPix 层级的划分影响到数据分布的均衡性及证认结果的准确性，为了更严谨地判断出层级划分对算法影响的优劣，我们把 $F1$ 值作为 HEALPix 层级划分的评价指标，$F1$ 值由精确率 (precision, P 值) 和召回率 (recall, R 值) 共同计算得出，简要介绍如下：

(1) 精确率。它表示的是预测为正的样本中有多少是真正的正样本，公式中包含将正类预测为负类的数量 (true positive, TP)，将负类预测为正类的数量 (false positive, FP)，计算公式为：

$$P = \frac{TP}{TP + FP} \quad . \tag{1}$$

(2) 召回率。它表示的是样本中的正例有多少被预测正确，公式中包含将正类预测为负类的数量 (false negative, FN)，计算公式为：

$$R = \frac{TP}{TP + FN} \quad . \tag{2}$$

(3) 综合评价指标 (F-Measure)。P 和 R 指标有时候会出现矛盾的情况，这样就需要综合考虑他们，最常见的方法就是 F-Measure (又称为 F-Score)。F-Measure 是 Precision 和 Recall 加权调和平均：

$$F1 = \frac{(a^2+1)P \times R}{a^2(P+R)} \quad . \tag{3}$$

当参数 $a = 1$ 时，就是最常见的 $F1$，也就是我们本次实验中用的 $F1$ 值，如下：

$$F1 = \frac{2 \times P \times R}{(P+R)} \quad . \tag{4}$$

由公式可知 $F1$ 综合了 P 和 R 的结果，当 $F1$ 较高时，则说明实验方法比较有效。

由图 11 可知，HEALPix 层级越低，划分的天区越大，落入同一天区的星体数目越多，匹配时易发生混淆导致匹配错误；HEALPix 层级越高，划分的天区越小，过小的天区易使星体跨越边界落入相邻分区中，导致本来可以匹配的星体因分区 ID 不同无法匹配，从而丢失数据。所以在 HEALPix 取 17 级时虽然精确率可达 100%，但召回率极低，说明星表数据在天区划分过细的情况下会大量丢失证认结果，对于时序重构来说，证认结果的丢失势必会极大影响光变曲线的生成；而在 13 级时 $F1$ 值最高，所以我们选择 HEALPix 层级为 13 级，可以使算法达到最优。

4.2.3 四种减少距离计算的证认方法评价指标对比

为了验证本文提出的快速证认方法的有效性，下面分别对四种可减少距离计算的证认方法进行 $F1$ 值的对比，4 种具体证认方法如下：

(1) 证认星表间相同 HEALPix 块号的星体若数目相同，直接经范围过滤后匹配输出，在范围内的星体认为匹配成功；当相同块号内星体数目不等时，在范围内的星体直接匹配，多出的星体证认失败。

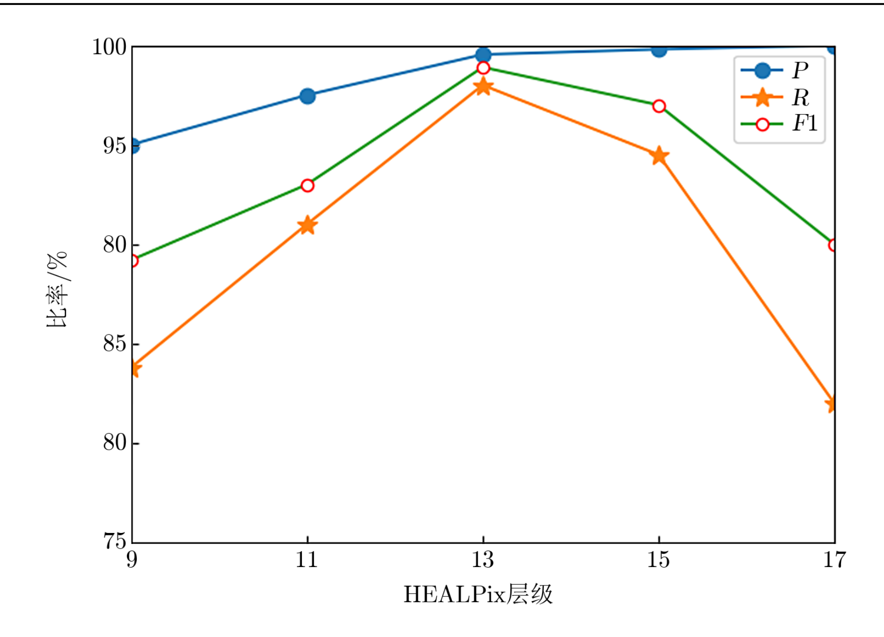


图 11　不同 HEALPix 层级的评价指标对比

(2) 证认星表间相同 HEALPix 块号的星体若数目相同，直接经范围过滤后匹配输出，在范围内的星体认为匹配成功；两块内星体数目不等时，两个 HEALPix 块标记为特殊区域，两块内星体严格距离证认，但不进行周围块的标记和证认。

(3) 证认星表间相同 HEALPix 块号的星体若数目相同，直接经范围过滤后匹配输出，在范围内的星体认为匹配成功；两块内星体数目不等时，两个 HEALPix 块标记为特殊区域，两块内星体严格距离证认，且进行周围块的标记和证认。

(4) 证认星表间相同 HEALPix 块号的星体若数目相同且为 1 时，对比两颗星体的亮度，亮度小于亮度阈值的星体匹配成功；若星体数目相同且大于 1 时，经范围过滤和亮度比较后留下的星体认为匹配成功。两块内星体数目不等时或对应星体的亮度大于亮度阈值时，两个 HEALPix 块标记为特殊区域，两块内星体严格距离证认，且进行周围块的标记和证认。

由图 12 可知，方法 (1) 性能最差，原因是没有考虑边界漏源问题及一对多或多对一星体证认问题，且直接通过块号匹配导致召回率和准确率很低。而方法 (4) 则根据相同块号内星体数目相同或不同采取不同证认方式，融合了星等信息，加入范围过滤和对边界漏源的考虑，极大地提高了算法的性能，所以本文的快速证认算法采用了方法 (4)。

4.2.4　快速证认算法与传统证认算法的速率及准确率对比测试

为了显示快速证认算法的性能，我们选取了 AST3 拍摄的 HD8500 文件的部分数据作为测试数据，在采用一个节点的情况下与传统证认算法进行了对比分析。图 13 显示求出所需要的时间，我们发现相同数据量下快速证认算法所用的计算时间明显小于传统的证认算法。

图 14 是两种算法所得出证认结果数据量的对比，发现两种算法的计算结果数据量相差无几。我们以经过严格距离计算的传统证认方法得出的证认结果为标准，再与快速证认算法

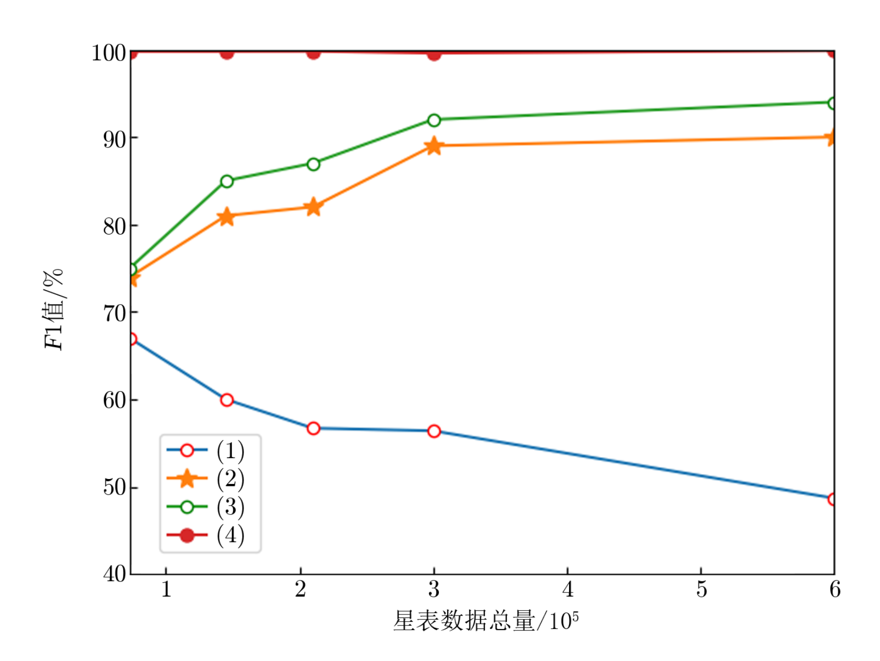


图 12 不同证认方法的 $F1$ 值对比

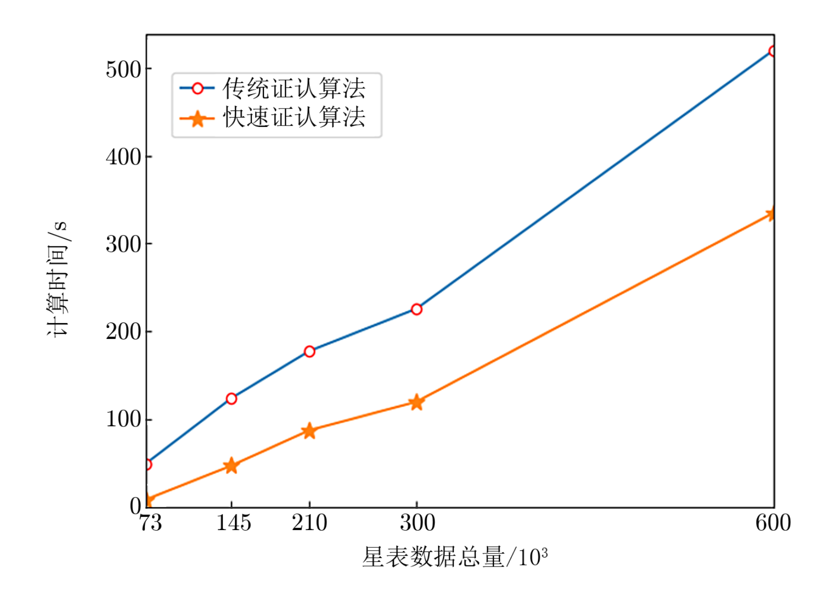


图 13 两种算法计算时间对比

得出的结果进行比较。如图 15 所示，蓝色曲线为快速证认算法的准确率曲线，可以看出快速证认算法的准确率极高，都在 99.5% 以上。

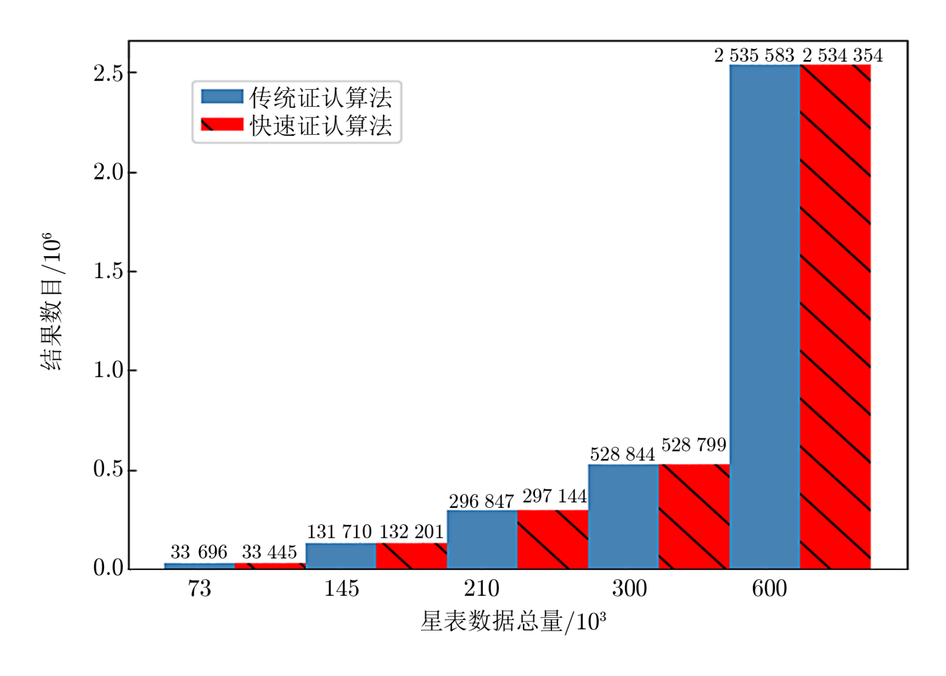


图 14　两种算法计算结果数目对比

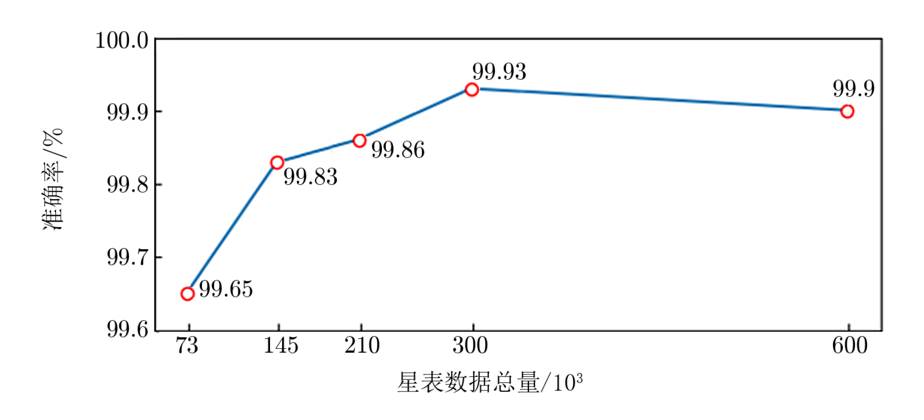


图 15　快速证认算法的准确率

从前面的对比可知，相比传统证认算法，快速证认算法加入了对星体的范围过滤和亮度比较，以及运用了免距离的证认算法，在保证了一定准确率的基础上，避免了大量的距离计算，大大减少了证认计算的时间，提高了证认的效率，可以更快得出证认计算结果。

4.3 MongoDB 的读写性能测试

4.3.1 MongoDB 的索引性能对比测试

MongoDB 的地理空间索引在查询地理数据时效率更高，下面对比了 MongoDB 的 2D 平面索引、2Dsphere 索引以及在 HEALPixID 这一列建立的 B 树索引查询数据的效率。

表 2 MongoDB 索引对比表

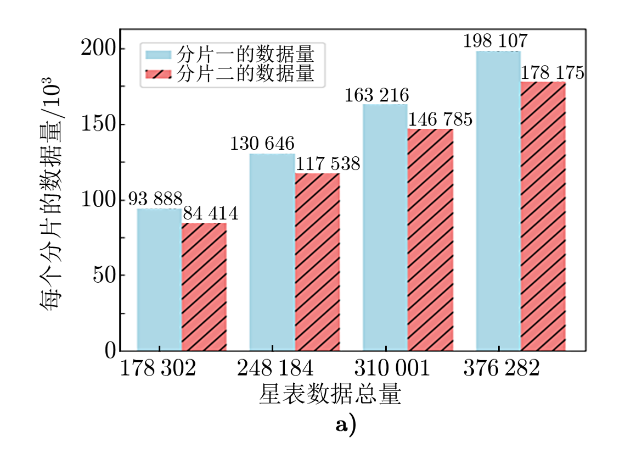
索引	星表数据查询量	时间/s		
		2D 索引	B 树索引	2Dsphere 索引
1	5.0×10^4	14.163	14.257	12.135
2	1.5×10^5	41.388	53.332	44.993
3	3.0×10^5	66.244	83.341	76.342
4	6.0×10^5	142.424	199.42	160.42
5	7.5×10^5	202.432	225.576	243.345

由表 2 可以看出，在数据量较大时，2Dsphere 球面索引的效率明显要低于 2D 平面索引的效率。所以我们对星表数据的赤经赤纬建立 2D 索引。

4.3.2 哈希分区的数据均衡测试

哈希函数会使数据随机分片，我们可以利用哈希分区这一静态负载均衡的特性将星表数据在一定范围内均衡地分布在不同的分片节点上。下面进行了在星表数据的 HEALPixID 这一列建立哈希索引和 B 树索引作为片键的两种情况下数据分布对比测试。

从图 16 我们可以看出，建立普通的 B 树索引会导致数据分布不均，节点间存储数据数目差距巨大；而哈希分区的负载均衡效果十分明显，可以使数据在所有节点间达到基本的均匀分布，从而分散不同节点对数据的读写压力。



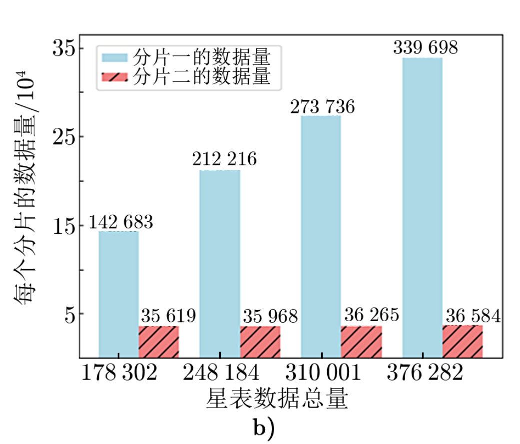


图 16 哈希索引 (图 a)) 和 B 树索引 (图 b)) 作为片键的数据分布对比

4.4 综合测试

传统时序数据生成方法是将星表文件导入关系型数据库，并通过交叉证认的结果对不同时间段的星体进行 MatchID 标记，再匹配数据库中相同 ID 标记的星体[19]，然后将结果导入到文件中，最后可以根据这些文件生成时序数据。本实验选取了 AST3 拍摄的 HD88500 部分星表文件作为测试数据，每个星表文件的数据量基本都在 29 000 ～ 39 000 的区间内，下面对本文方法与传统时序数据生成方法作了对比测试。

根据表 3 中的实验结果可以看出，基于 MongoDB 的快速证认算法显然比一般传统的方法更加高效，而且随着数据量的增大，效果越明显。

表 3　传统证认方法与快速证认方法的比较

属性	HD88500				
星表文件数目/个	16	22	30	60	120
传统证认方法/min	6.30	9.44	17.01	38.15	87.40
快速证认方法/min	4.18	6.20	10.94	21.32	48.07

5 总　结

本文设计了一种基于 MongoDB 的海量天文星表数据的快速时序重构方法。一方面利用 MongoDB 的文档型存储和地理空间索引提高了存储和查询数据的效率；另一方面通过特定区域的范围过滤、星等信息比较及局部证认计算代替全局计算，来减少传统交叉证认方法中的距离计算量。实验结果证明，本文在数据的访存优化、证认算法效率等方面取得了一定的改进效果，特别是对于海量同源星表的连续时间上的证认问题，本研究的快速时序重构方法具有良好的应用价值，可以有效解决大规模天文星表的批量时序重构的访存、计算效率等问题，从而推动时域天文学时代星体光变曲线相关研究。

参考文献：

[1] 高丹, 张彦霞, 赵永恒. 天文研究与技术, 2005, 2(3): 186
[2] 高丹. 博士论文. 北京: 中国科学院国家天文台, 2008: 1
[3] Zhao Q, Sun J, Yu C, et al. ICA3PP, 2009: 604
[4] Zhao Q, Sun J, Yu C, et al. Transactions of Tianjin University, 2011, 17(1): 62
[5] 徐洋, 吴潮, 万萌, 等. 天文研究与技术, 2013, 10(3): 273
[6] 徐洋. 硕士论文. 湖北: 三峡大学, 2013: 1
[7] 杜鹏. 硕士论文. 山东: 山东大学, 2013: 1
[8] Du P, Ren J J, Pan J C, et al. Science China: Physics Mechanics & Astronomy, 2014, 57 (3): 577
[9] 万萌. 博士论文. 北京: 中国科学院国家天文台, 2016: 1
[10] 杨晨, 翁祖建, 孟小峰, 等. 计算机研究与发展, 2017, 54(2): 248

[11] 冯磊, 杨昭颖, 李文吉, 等. 地质学刊, 2019, 43(03): 421
[12] 程江林. 硕士论文. 安徽: 安徽大学, 2020: 1
[13] 周尧, 刘超, 徐树楠, 等. 测绘与空间地理信息, 2018, 41(09): 71
[14] Li K, Yu C, Tang S, et al. 15th IEEE International Symposium on Parallel and Distributed Processing with Applications. Guangzhou, China, 2017: 1074
[15] 任明飞, 李学军, 崔蒙蒙, 等. 电脑知识与技术, 2019, 15(34): 1
[16] 徐思路. 硕士论文. 辽宁: 大连海事大学, 2018: 1
[17] Kristina Chodorow. MongoDB 权威指南 (第 2 版). 邓强, 王明辉, 译. 北京: 人民邮电出版, 2014: 121
[18] 张雯杰, 蔡佳玲编著. MongoDB 从入门到商业实战. 北京: 北京电子出版社, 2019: 47
[19] 熊聪聪, 付立艳, 赵青. 计算机应用与软件, 2021, 38(04): 17

Research on Fast Time Series Reconstruction of Massive Astronomical Catalog Data Based on MongoDB

XU Dan-ying, ZHAO Qing, QUAN Wen-li, SONG Hong-zhuang

(College of Artificial Intelligence, Tianjin University of Science & Technology, Tianjin 300457, China)

Abstract: The explosive growth of astronomical data leads to low efficiency in the generation of astronomical time series data by traditional scientific calculation methods, which directly affects the scientific output of time domain astronomy. In this paper, we propose a fast method to authenticate the same catalog and a MongoDB-based application to reduce the distance computation. In order to solve the efficiency problem of batch time series reconstruction of large-scale astronomical catalogues, we focus on the optimization of original data access and the improvement of authentication computation speed.

The experimental results show that this method can improve the efficiency of time series data generation more effectively than the traditional multi-band cross-validation algorithm and relational database method, it provides a new idea for the reconstruction of time series and the generation of light curves for the large-scale catalogue data of the time-domain astronomical time-frequency sampling telescope.

Key words: MongoDB; geospatial index; memory access optimization; authentication optimization